	Type	L #	Hits	Search Text	DBs	Time Stamp
1	IS&R	L1	1262	(712/216-218).CCLS.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:11
2	BRS	L2	565	1 and schedul$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:12
3	BRS	L3	88	2 and register adj allocat$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:14
4	BRS	L4	25006774	@ad<"20031212"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:14
5	BRS	L5	83	3 and 4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:14
6	BRS	L6	4	4 and topological adj dependenc$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:26

7	BRS	L7	6	topological adj dependenc$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:18

	Type	L #	Hits	Search Text	DBs	Time Stamp
8	BRS	L8	2754	4 and instruction near3 dependenc$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:19
9	BRS	L9	56	5 and instruction near3 dependenc$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:25
10	BRS	L10	0	9 and scratch adj register	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:20
11	BRS	L11	0	9 and scratch$ adj register$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:21
12	BRS	L12	633	scratch$ adj register$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:21
13	BRS	L13	450	scratch adj register	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:21

14	BRS	L14	112	preserved adj register	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:21

	Type	L #	Hits	Search Text	DBs	Time Stamp
15	BRS	L15	5	13 and 14	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:21
16	BRS	L16	0	9 and 15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:22
17	BRS	L17	5336	schedul$ near3 instruction$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:23
18	BRS	L18	0	17 nad allocat$ near5 register$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:24
19	BRS	L19	861	17 and allocat$ near5 register$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:24
20	BRS	L20	760	19 and 4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:25

21	BRS	L21	518	20 and dependenc$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:25

	Type	L #	Hits	Search Text	DBs	Time Stamp
22	BRS	L22	361	20 and instruction near3 dependenc$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:25
23	BRS	L23	0	22 and topological adj dependenc$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:27
24	BRS	L24	1	22 and (live adj range near3 operand) and dependenc$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 10:27

	Type	L #	Hits	Search Text	DBs	Time Stamp
1	BRS	L1	2340	instruction adj schedul$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 17:57
2	BRS	L2	2225	register adj allocat$	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 17:57
3	BRS	L3	392	1 and 2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 17:57
4	BRS	L4	25006774	@ad<"20031212"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 17:58
5	BRS	L5	347	3 and 4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 17:58
6	BRS	L6	0	5 and deoendence	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 17:59

7	BRS	L7	97	5 and dependence	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:26

	Type	L #	Hits	Search Text	DBs	Time Stamp
8	BRS	L8	7	7 and function adj call	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:26
9	BRS	L9	1	7 and function adj call and scratch adj register	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:04
10	BRS	L10	0	7 and function adj call and scratch adj register and preserved adj register	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:26
11	BRS	L11	0	717/140,153,161	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:24
12	IS&R	L12	810	(717/140,153,161).CCLS.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:24
13	BRS	L13	709	12 and 4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:25

14	BRS	L14	104	13 and 1	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:25

	Type	L #	Hits	Search Text	DBs	Time Stamp
15	BRS	L15	110	13 and 2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:25
16	BRS	L16	104	14 and 1	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:26
17	BRS	L17	60	16 and 2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:26
18	BRS	L18	33	17 and dependence	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:26
19	BRS	L19	0	18 and function adj call and scratch adj register and preserved adj register	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:26
20	BRS	L20	3	18 and function adj call	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2006/06/10 18:26

Terms used **allocation** and **register** and **scrach register** and **preserved register** and **live** and **range** and **topological adn dependence** and **operand** and **scheduling**

Found **8,124** of **177,263**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown
Relevance scale

1 Parallelizing nonnumerical code with selective scheduling and software pipelining
Soo-Mook Moon, Kemal Ebcioğlu
November 1997 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 19 Issue 6
Publisher: ACM Press
Full text available: pdf(543.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Instruction-level parallelism (ILP) in nonnumerical code is regarded as scarce and hard to exploit due to its irregularity. In this article, we introduce a new code-scheduling technique for irregular ILP called "selective scheduling" which can be used as a component for superscalar and VLIW compilers. Selective scheduling can compute a wide set of independent operations across all execution paths based on renaming and forward-substitution and can compute availab ...

Keywords: VLIW, global instruction scheduling, instruction-level parallelism, software pipelining, speculative code motion, superscalar

2 Software pipelining showdown: optimal vs. heuristic methods in a production compiler
John Ruttenberg, G. R. Gao, A. Stoutchinin, W. Lichtenstein
May 1996 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation PLDI '96**, Volume 31 Issue 5
Publisher: ACM Press
Full text available: pdf(1.43 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper is a scientific comparison of two code generation techniques with identical goals --- generation of the best possible software pipelined code for computers with instruction level parallelism. Both are variants of *modulo scheduling*, a framework for generation of software pipelines pioneered by Rau and Glaser [RaG181], but are otherwise quite dissimilar.One technique was developed at Silicon Graphics and is used in the MIPSpro compiler. This is the production compiler for SGI's s ...

3 An efficient resource-constrained global scheduling technique for superscalar and VLIW processors
Soo-Mook Moon, Kemal Ebcioğlu

December 1992 **ACM SIGMICRO Newsletter , Proceedings of the 25th annual international symposium on Microarchitecture MICRO 25**, Volume 23 Issue 1-2
Publisher: IEEE Computer Society Press, ACM Press
Full text available: pdf(2.05 MB) Additional Information: full citation, references, citings, index terms

Keywords: VLIW, compile-time parallelization, instruction-level parallelism, superscalar

4 Stage scheduling: a technique to reduce the register requirements of a modulo schedule
Alexandre E. Eichenberger, Edward S. Davidson
December 1995 **Proceedings of the 28th annual international symposium on Microarchitecture**
Publisher: IEEE Computer Society Press
Full text available: pdf(1.24 MB) Additional Information: full citation, references, citings, index terms

5 The Marion system for retargetable instruction scheduling
David G. Bradlee, Robert R. Henry, Susan J. Eggers
May 1991 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1991 conference on Programming language design and implementation PLDI '91**, Volume 26 Issue 6
Publisher: ACM Press
Full text available: pdf(1.35 MB) Additional Information: full citation, references, citings, index terms

6 A Dynamic Programming Approach to Optimal Integrated Code Generation
Christoph Keßler, Andrzej Bednarski
August 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN workshop on Languages, compilers and tools for embedded systems LCTES '01 , Proceedings of the 2001 ACM SIGPLAN workshop on Optimization of middleware and distributed systems OM '01**, Volume 36 Issue 8
Publisher: ACM Press
Full text available: pdf(216.58 KB) Additional Information: full citation, abstract, references, citings, index terms

Phase-decoupled methods for code generation are the state of the art in compilers for standard processors but generally produce code of poor quality for irregular target architectures such as many DSPs. In that case, the generation of efficient code requires the simultaneous solution of the main subproblems instruction selection, instruction scheduling, and register allocation, as an integrated optimization problem.

In contrast to compilers for standard processors, code generation for ...

Keywords: dynamic programming, instruction scheduling, instruction selection, integrated code generation, register allocation, time profile

7 Software pipelining
Vicki H. Allan, Reese B. Jones, Randall M. Lee, Stephen J. Allan
September 1995 **ACM Computing Surveys (CSUR)**, Volume 27 Issue 3
Publisher: ACM Press

Full text available: pdf(4.72 MB) Additional Information: full citation, abstract, references, citings, index terms

Utilizing parallelism at the instruction level is an important way to improve performance. Because the time spent in loop execution dominates total execution time, a large body of optimizations focuses on decreasing the time to execute each iteration. Software pipelining is a technique that reforms the loop so that a faster execution rate is realized. Iterations are executed in overlapped fashion to increase parallelism.Let {ABC}n
Keywords: instruction level parallelism, loop reconstruction, optimization, software pipelining

8 Accurate and efficient predicate analysis with binary decision diagrams

John W. Sias, Wen-Mei W. Hwu, David I. August
December 2000 **Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture**
Publisher: ACM Press
Full text available: pdf(336.92 KB) ps(6.25 MB) Additional Information: full citation, references, citings, index terms
Publisher Site

9 Efficient superscalar performance through boosting

Michael D. Smith, Mark Horowitz, Monica S. Lam
September 1992 **ACM SIGPLAN Notices , Proceedings of the fifth international conference on Architectural support for programming languages and operating systems ASPLOS-V**, Volume 27 Issue 9
Publisher: ACM Press
Full text available: pdf(1.63 MB) Additional Information: full citation, abstract, references, citings, index terms

The foremost goal of superscalar processor design is to increase performance through the exploitation of instruction-level parallelism (ILP). Previous studies have shown that speculative execution is required for high instruction per cycle (IPC) rates in non-numerical applications. The general trend has been toward supporting speculative execution in complicated, dynamically-scheduled processors. Performance, though, is more than just a high IPC rate; it also depends upon instruction count ...

10 VLIW compilation techniques in a superscalar environment

Kemal Ebcioglu, Randy D. Groves, Ki-Chang Kim, Gabriel M. Silberman, Isaac Ziv
June 1994 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1994 conference on Programming language design and implementation PLDI '94**, Volume 29 Issue 6
Publisher: ACM Press
Full text available: pdf(1.30 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe techniques for converting the intermediate code representation of a given program, as generated by a modern compiler, to another representation which produces the same run-time results, but can run faster on a superscalar machine. The algorithms, based on novel parallelization techniques for Very Long Instruction Word (VLIW) architectures, find and place together independently executable operations that may be far apart in the original code. i.e., they may be se ...

Keywords: VLIW, compiler optimizations, global scheduling, profiling directed feedback, software pipelining, superscalars

11 An experimental study of several cooperative register allocation and instruction scheduling strategies

Cindy Norris, Lori L. Pollock

December 1995 **Proceedings of the 28th annual international symposium on Microarchitecture**

Publisher: IEEE Computer Society Press

Full text available: pdf(1.17 MB) Additional Information: full citation, references, citings, index terms

12 Real-time shading

Marc Olano, Kurt Akeley, John C. Hart, Wolfgang Heidrich, Michael McCool, Jason L. Mitchell, Randi Rost

August 2004 **Proceedings of the conference on SIGGRAPH 2004 course notes GRAPH '04**

Publisher: ACM Press

Full text available: pdf(7.39 MB) Additional Information: full citation, abstract

Real-time procedural shading was once seen as a distant dream. When the first version of this course was offered four years ago, real-time shading was possible, but only with one-of-a-kind hardware or by combining the effects of tens to hundreds of rendering passes. Today, almost every new computer comes with graphics hardware capable of interactively executing shaders of thousands to tens of thousands of instructions. This course has been redesigned to address today's real-time shading capabili ...

13 Register allocation for predicated code

Alexandre E. Eichenberger, Edward S. Davidson

December 1995 **Proceedings of the 28th annual international symposium on Microarchitecture**

Publisher: IEEE Computer Society Press

Full text available: pdf(1.31 MB) Additional Information: full citation, references, citings, index terms

Keywords: hyperblocks, interference, predicated execution, register allocation, software pipelining

14 The Jalapeño dynamic optimizing compiler for Java

Michael G. Burke, Jong-Deok Choi, Stephen Fink, David Grove, Michael Hind, Vivek Sarkar, Mauricio J. Serrano, V. C. Sreedhar, Harini Srinivasan, John Whaley

June 1999 **Proceedings of the ACM 1999 conference on Java Grande**

Publisher: ACM Press

Full text available: pdf(1.34 MB) Additional Information: full citation, references, citings, index terms

15 Compiler technology for parallel machines: Register allocation for optimal loop scheduling

Qi Ning

October 1993 **Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: distributed computing - Volume 2**

Publisher: IBM Press

Full text available: pdf(1.03 MB) Additional Information: full citation, abstract, references

One of the major challenges in designing optimizing compilers, especially for scientific

computation, is to take advantage of the parallelism in loops in order to obtain maximum speedup on parallel computer architectures. Optimal loop scheduling is therefore one of the most important topics studied by many computer scientists. However, how to allocate a minimum number of registers to support optimal loop scheduling for parallel architectures is less understood. In this report, we propose a simul ...

16 Space-time scheduling of instruction-level parallelism on a raw machine

Walter Lee, Rajeev Barua, Matthew Frank, Devabhaktuni Srikrishna, Jonathan Babb, Vivek Sarkar, Saman Amarasinghe

October 1998 **ACM SIGPLAN Notices , ACM SIGOPS Operating Systems Review , Proceedings of the eighth international conference on Architectural support for programming languages and operating systems ASPLOS-VIII**, Volume 33 , 32 Issue 11 , 5

Publisher: ACM Press

Full text available: pdf(1.79 MB) Additional Information: full citation, abstract, references, citings, index terms

Increasing demand for both greater parallelism and faster clocks dictate that future generation architectures will need to decentralize their resources and eliminate primitives that require single cycle global communication. A Raw microprocessor distributes all of its resources, including instruction streams, register files, memory ports, and ALUs, over a pipelined two-dimensional mesh interconnect, and exposes them fully to the compiler. Because communication in Raw machines is distributed, com ...

17 Parallel processing: a smart compiler and a dumb machine

Joseph A. Fisher, John R. Ellis, John C. Ruttenberg, Alexandru Nicolau

June 1984 **ACM SIGPLAN Notices , Proceedings of the 1984 SIGPLAN symposium on Compiler construction SIGPLAN '84**, Volume 19 Issue 6

Publisher: ACM Press

Full text available: pdf(1.05 MB) Additional Information: full citation, abstract, references, citings

Multiprocessors and vector machines, the only successful parallel architectures, have coarse-grained parallelism that is hard for compilers to take advantage of. We've developed a new fine-grained parallel architecture and a compiler that together offer order-of-magnitude speedups for ordinary scientific code.

18 Fusion-based register allocation

Guei-Yuan Lueh, Thomas Gross, Ali-Reza Adl-Tabatabai

May 2000 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 22 Issue 3

Publisher: ACM Press

Full text available: pdf(475.45 KB) Additional Information: full citation, abstract, references, citings, index terms, review

The register allocation phase of a compiler maps live ranges of a program to registers. If there are more candidates than there are physical registers, the register allocator must spill a live range (the home location is in memory) or split a live range (the live range occupies multiple locations). One of the challenges for a register allocator is to deal with spilling and splitting together. Fusion-based register allocation uses the structure of the program to make splitting and spilling d ...

Keywords: performance evaluation, register allocation

19 Sentinel scheduling: a model for compiler-controlled speculative execution

Scott A. Mahlke, William Y. Chen, Roger A. Bringmann, Richard E. Hank, Wen-Mei W. Hwu, B. Ramakrishna Rau, Michael S. Schlansker

November 1993 **ACM Transactions on Computer Systems (TOCS)**, Volume 11 Issue 4
Publisher: ACM Press

Full text available: pdf(2.26 MB) Additional Information: full citation, abstract, references, citings, index terms

Speculative execution is an important source of parallelism for VLIW and superscalar processors. A serious challenge with compiler-controlled speculative execution is to efficiently handle exceptions for speculative instructions. In this article, a set of architectural features and compile-time scheduling support collectively referred to as sentinel scheduling is introduced. Sentinel scheduling provides an effective framework for both compiler-controlled speculative executi ...

Keywords: VlIW processor, exception detection, exception recovery, instruction scheduling, instruction-level parallelism, speculative execution, superscalar processor

20 Register allocation over the program dependence graph

Cindy Norris, Lori L. Pollock
June 1994 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1994 conference on Programming language design and implementation PLDI '94**, Volume 29 Issue 6
Publisher: ACM Press

Full text available: pdf(1.27 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes RAP, a Register Allocator that allocates registers over the Program Dependence Graph (PDG) representation of a program in a hierarchical manner. The PDG program representation has been used successfully for scalar optimizations, the detection and improvement of parallelism for vector machines, multiple processor machines, and machines that exhibit instruction level parallelism, as well as debugging, the integration of different versions of a program, and translation of ...